REST

Service Application Development

Šta je REST?

- Representational State Transfer ili skraćeno REST je koncept koji je zasnovan na veb-standardnima i HTTP protokolu. REST je prvi put definisao 2000. godine Roj Filding. Centralni deo REST arhitekture zauzima resurs. Resursu se pristupa preko zajedničkog interfejsa uz korišćenje HTTP metoda. Stoga se u ovakvoj arhitekturi ne koristi XML, a postiže se identična platformska i jezička nezavisnost i otpornost na firewall. REST omogućava resursima da imaju različitu reprezentaciju, kao što je na primer tekst, XML, JSON i tako dalje

Arhitektura REST-a

- REST koncept koristi URL-ove za reprezentaciju objekata i HTTP metod za rukovanje njima. Najpoznatije metode su GET za preuzimanje i POST za kreiranje novih podataka
- Za ažuriranje podataka koristi se HTTP metod PUT
- DELETE HTTP metod možemo da koristimo za brisanje. Parametre prosleđujemo kroz URL String, pa je jedino važno prilikom korišćenja proveriti HTTP metod. Ukoliko je metod DELETE, brišemo po ključu prosleđenom kroz URL. Ukoliko nije, biće izvršena neka druga operacija (najverovatnije preuzimanje podataka za zadati parametar)

Metoda	Opis
GET	čita resurs
PUT	kreira resurs
POST	uklanja resurs
DELETE	ažurira postojeći ili kreira novi resurs

RESTFul veb-servisi

- Veb-servisi koji se baziraju na upravo opisanoj REST arhitekturi mogu se nazvati RESTFul veb-ervisima.
- RESTFul veb-servisi uglavnom definišu URI na kojem se servis nalazi, podržane tipove, koji mogu biti XML, text, JSON ili pak korisnički definisani i na kraju set operacija koje će podržavati servis

RESTFul i Python

- Iako je RESTful arhitekturu moguće ostvariti i pomoću standardnih Python biblioteka, postoji mnoštvo okvira za implementaciju ove arhitekture

Flask-RESTful

Flask-Restless

django REST framework

REST server implementacija

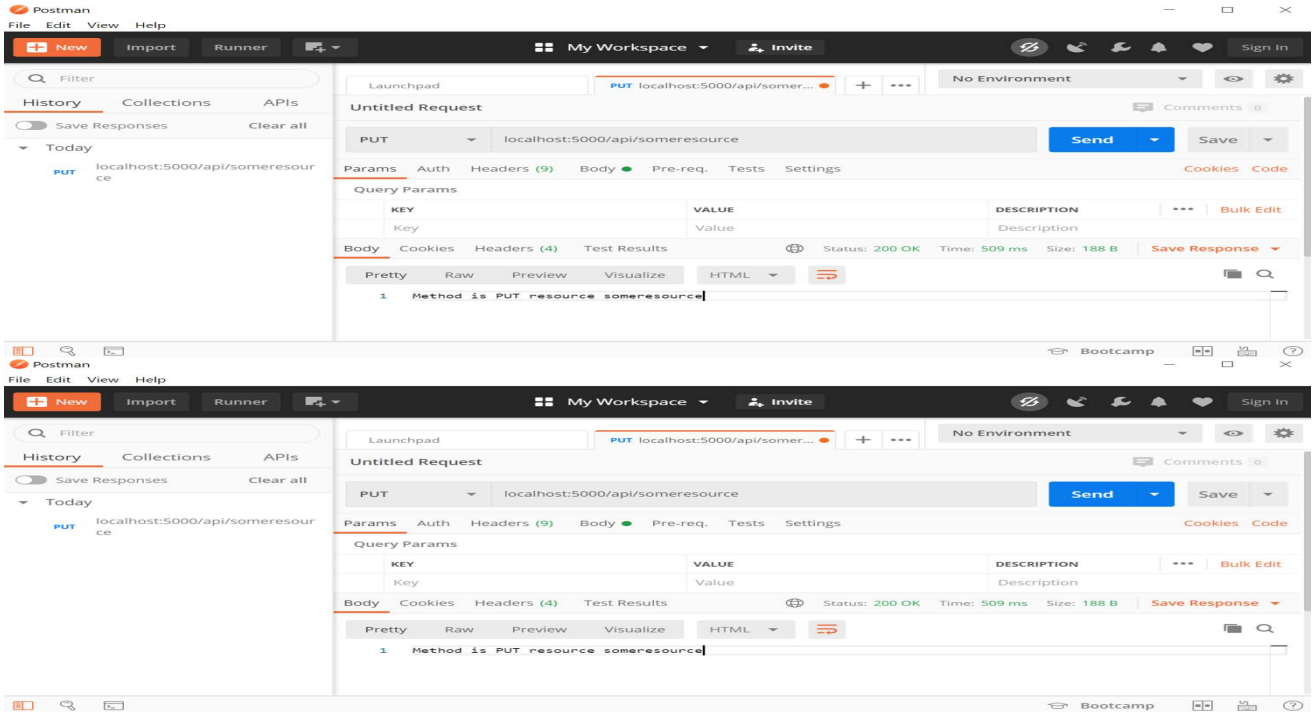
(sad-ex03 flaskrequest.py)


```python
from flask import Flask, request

app = Flask(__name__)

@app.route("/api/<resource>",methods=['GET','POST','PUT','DELETE'])
def req(resource):
    if request.method.lower() == "get":
        return f"Method is GET for resource {resource}"
    elif request.method.lower() == "post":
        return f"Method is POST resource {resource}"
    elif request.method.lower() == "put":
        return f"Method is PUT resource {resource}"
    elif request.method.lower() == "delete":
        return f"Method is DELETE resource {resource}"

app.run()
```

Postman

https://www.postman.com/

- Postman je neizostavni alat prilikom testiranja REST servisa



REST server implementacija

(sad-ex03 flaskrequest.py)


```python
from flask import Flask, request

app = Flask(__name__)

@app.route("/api/<resource>",methods=['GET','POST','PUT','DELETE'])
def req(resource):
    if request.method.lower() == "get":
        return f"Method is GET for resource {resource}"
    elif request.method.lower() == "post":
        return f"Method is POST resource {resource}"
    elif request.method.lower() == "put":
        return f"Method is PUT resource {resource}"
    elif request.method.lower() == "delete":
        return f"Method is DELETE resource {resource}"

app.run()
```

Flask RESTful

https://flask-restful.readthedocs.io/en/latest/

FlaskRESTful

pip install flask-restful

```
from flask import Flask
from flask_restful import Resource, Api

app = Flask(__name__)
api = Api(app)

class MyResource(Resource):
    def get(self):
        return {'firstname':'Jason','lastname':'Momoa'}

api.add_resource(MyResource, '/api/user')
```

Flask RESTful

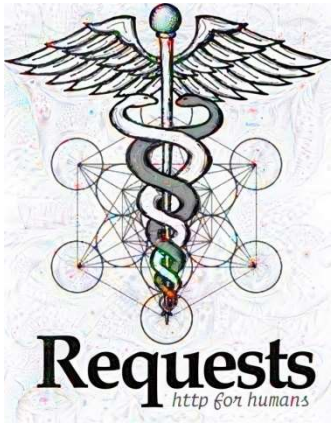
https://flask-restful.readthedocs.io/en/latest/

FlaskRESTful

```python
class Calc(Resource):
    def get(self,a,b):
        return {"res":a+b}
    def post(self):
        data = request.get_json(force=True)
        return data["a"] + data["b"]

api.add_resource(Calc,"/calc","/calc/<int:a>/<int:b>")
```

Requests

https://requests.readthedocs.io/en/master/



pip install requests

```
import requests
res = requests.get("https://samples.openweathermap.org/data/2.5/weather?id=2172797&
appid=439d4b804bc8187953eb36d2a8c26a02")
res = res.json()
print(f"Weather for {res['name']}")
print(f"Temperature: {res['main']['temp']}")
print(f"Pressure: {res['main']['pressure']}")
print(f"Humidity: {res['main']['humidity']}")
```

```
import requests
req = requests.post(
    "http://localhost:5000",
    json={"marco":"polo"}
)
print(req.status_code)
print(req.text)
```

```
import requests
req = requests.post(
    "http://localhost:5000",
    data={"marco":"polo"}
)
print(req.status_code)
print(req.text)
```

Urllib request

https://docs.python.org/3/library/urllib.request.html

```python
import urllib.request as request

req = request.Request("http://localhost:5000")
req.method = "POST"
req.add_header("Content-Type","application/x-www-form-urlencoded")
req.data = b"Marco=Polo"

res = request.urlopen(req)

print(res.read().decode())
```

```python
import urllib.request as request
import json
req = request.Request("http://localhost:5000")
req.method = "POST"
req.add_header("Content-Type","application/json")
req.data = json.dumps({"marco":"polo"}).encode()
res = request.urlopen(req)
print(res.read().decode())
```

```python
import urllib.request as request
import json
res = request.urlopen("https://samples.openweathermap.org/data/2.5/weather?id=2172797&
appid=439d4b804bc8187953eb36d2a8c26a02")
res = json.load(res)
print(f"Weather for {res['name']}")
print(f"Temperature: {res['main']['temp']}")
print(f"Pressure: {res['main']['pressure']}")
print(f"Humidity: {res['main']['humidity']}")
```

```
Weather for Cairns
Temperature: 300.15
Pressure: 1007
Humidity: 74
```

Obezbeđivanje REST servisa

- REST servisi nisu predviđeni da čuvaju stanje ali je upotreba sesija moguća
- Pristup REST servisima ograničava se ključevima
- U novijim implementacijama, u okviru REST-a česta je upotreba JWT tokena

JWT – JSON Web Token
(sad-ex03 simplejwt.py)

- JWT je tehnika u kojoj se klijentu izdaje potpisani token sa podacima
- Klijent daje token na validaciju prilikom svake autorizacije
- Token je validan ukoliko njegov potpis odgovara potpisu napravljenom prilikom validacije

pip install pyjwt

```
key = "secret_key"
data = {
    "role":"admin",
    "exp":time.time()+30
}
token = jwt.encode(data,key,'HS256')
```

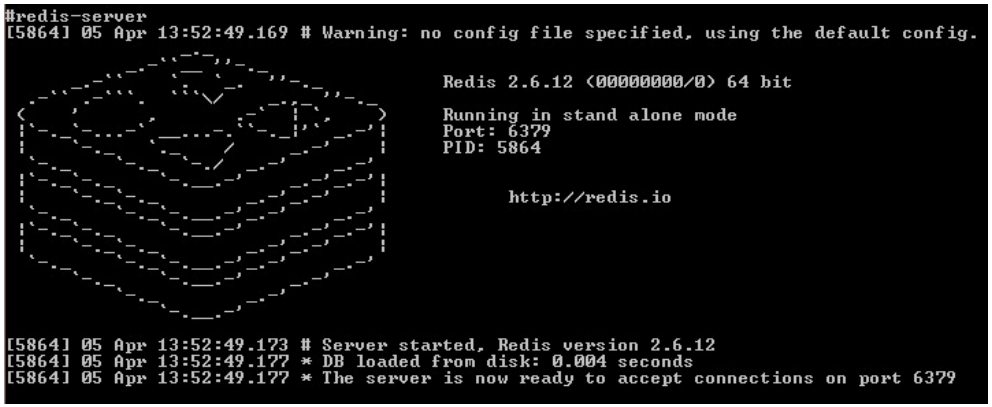
```
edata = jwt.decode(token,key,'HS256')
```

Redis (http://redis.io/)

- Redis je key-value baza podataka koja podatke čuva u memoriji
- Redis se najčešće koristi kao
  - Sistem za keširanje
  - Sistem za skladištenje sesija
  - Sistem za privremeno sinhronizaciju
  - Message queue-ing sistem

Upotreba redisa

- Redis zahteva runtime (server) da bi mogao biti korišćen



- Direktan konzolni pristup redis serveru vrši se pomoću alata **redis-cli**
- Ključevi u redisu se postavljaju / preuzimaju metodama **set** i **get**

```
#redis-cli
redis 127.0.0.1:6379> set marco polo
OK
redis 127.0.0.1:6379> get marco
"polo"
redis 127.0.0.1:6379>
```

Povezivanje redisa sa Python-om

- Biblioteka za rad sa redis bazom podataka dostupna je na pypi repozitorijumu

pip install redis

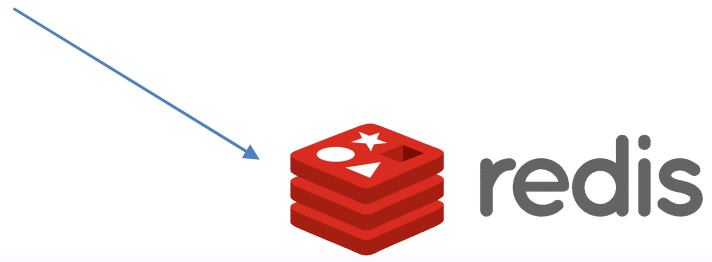
```
import redis

r = redis.Redis()
r.set("peter","Peter Jackson")
```

Perzistentnost objekata pomoću Redisa
(sad-ex03 simpleredis.py)

- Redis ne može direktno prihvatiti objekte, ali se oni mogu serijalizovati prilikom smeštanja u Redis

```
user = {
    "firstname":"Peter",
    "lastname":"Jackson"
}
r.set("peter",json.dumps(user))
```

```
user = json.loads(r.get("peter"))
print(
    user["firstname"],
    user["lastname"]
)
```



Redis publisher subscriber
(sad-ex03 redispubsub.py)

- Redis podržava publisher / subscriber model

```
r = redis.Redis()
r.publish("messages","Hello!")
```

```
r = redis.Redis()
ps = r.pubsub()
ps.subscribe({"message"})
for msg in ps.listen():
    try:
        print(msg["data"].decode())
    except:
        pass
```

```
r = redis.Redis()
ps = r.pubsub()
ps.subscribe({"message"})
def main_loop():
    while True:
        msg = ps.get_message(ignore_subscribe_messages=True)
        if msg:
            print(msg["data"].decode())
        time.sleep(0.001)
threading.Thread(None,main_loop).start()
```

Redovi poruka

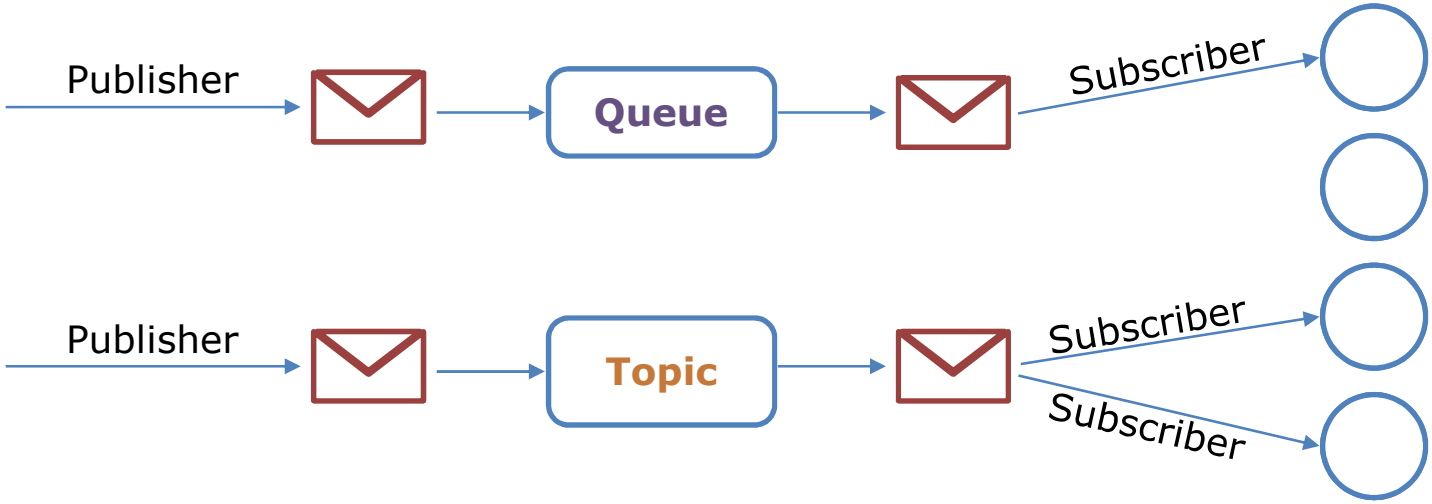
- Redovi poruka su aplikacije ili delovi aplikacija koji su u stanju da prihvataju, čuvaju i distribuiraju poruke

- Često se u svrhu redova poruka koriste namenski programi

ACTIVEMQ RabbitMQ ØMQ RocketMQ

Redovi i teme

- Dva osnovna načina distribucije poruka su red (**queue**) i tema (**topic**)
- Red uklanja dostavljenu poruku nakon prve uspešne isporuke
- Tema isporučuje poruku svim zainteresovanim učesnicima



Apache Active MQ

- ActiveMQ je popularan red poruka, otvorenog koda
- ActiveMQ redu poruka se može pristupiti mrežno nekim od protokola (REST, OpenWire, Stomp i drugi)
- ActiveMQ se može koristiti putem Python-a

pip install stomp.py

```
import stomp
```

Povezivanje i slanje poruka
(sad-ex03 stomp)

- Povezivanje

```
conn = stomp.Connection()
conn.connect('admin', 'admin', wait=True)
```

- Slanje poruke

```
conn.send(body=str(time.time()), destination='/queue/results')
```

- Prekidanje konekcije

```
conn.disconnect()
```

Preuzimanje poruka
(sad-ex03 stomp)

```
class MyListener(stomp.ConnectionListener):
    def on_error(self, headers, message):
        print('received an error "%s"' % message)
    def on_message(self, headers, message):
        print('received a message "%s"' % message)

conn = stomp.Connection()
conn.set_listener('', MyListener())
conn.connect(wait=True)

conn.subscribe(destination='/queue/results', id=1, ack='auto')
```

Web Socket
(sad-ex03 wschat.py)

- Omogućava perzistentnu vezu između klijenta i servera
- Veza je asinhrona (duplex)
- Tehnologija je podržana u okviru svih pregledača
- Postoje različite implementacije za Python

pip install websockets

https://websockets.readthedocs.io/en/stable/intro.html